Implementation of bio-inspired adaptive wavelet transforms in FPGAs. Modelling, validation and profiling of the algorithm.

Rubén Salvador, Félix Moreno, Teresa Riesgo
Centre of Industrial Electronics
Universidad Politécnica de Madrid
José Gutierrez Abascal, 2
28006, Madrid, Spain
Email: ruben.salvador@upm.es

Lukáš Sekanina
Faculty of Information Technology
Brno University of Technology
Božetěchova 2
612 66 Brno, Czech Republic
Email: sekanina@fit.vutbr.cz

***Abstract*—Providing embedded systems with adaptation capabilities is an increasing importance objective in design community. This work deals with the implementation of adaptive compression schemes in FPGA devices by means of a bio-inspired algorithm. A simplified version of an Evolution Strategy using fixed point arithmetic is proposed. Specifically, a simpler than the standard (hardware friendly) mutation operator is designed, modelled and validated using a high-level language. HW/SW partitioning issues are considered and code profiling accomplished to validate the proposal. Preliminary results of the proposed hardware architecture are also shown.**

I. Introduction

JPEG200, in contrast with the Discrete Cosine Transform (DCT) used in JPEG, is based on the Discrete Wavelet Transform (DWT) [1]. It is a very useful tool for (adaptive) image compression algorithms, since it provides a transform framework that can be adapted to the type of images being handled. This feature allows to improve the performance of the transform according to each particular type of image so that improved compression (in terms of size) can be achieved, depending on the wavelet used.

One of the current design challenges in embedded systems engineering is the implementation of adaptation capabilities. With previous compression algorithms this was not possible (from the point of view of the transform module in the algorithm), but, DWT opens up a possibility for this task to be tackled. Having a system able to adapt its compression performance according to the type of images being handled, may help in, for example, the calibration of image processing systems. Depending on where the system is deployed, certain tunings to the transform coefficients may help in increasing the quality of the transform, and, subsequently, the quality of the compression.

The accepted standard wavelet considered to be the state of the art in compression of photographic images is the hand-designed D9/7 Cohen-Daubechies-Feauveau (9/7-CDF or also D9/7). Therefore, though it has a better coding performance compared to JPEG, this will hold just for the type of images the wavelet is adapted to.

This work deals with the implementation of adaptive wavelet transforms in FPGA devices. An Evolutionary Algorithm (EA), specifically an Evolution Strategy (ES) [2] was chosen as the bio-inspired search algorithm.

Since the intended deployment platform is an FPGA device, a relatively low computing power will be available, what will, undoubtedly, affect the performance of the evolutionary search. However, this will provide the system with adaptation capabilities.

It has to be noted that most previous works on *adaptivity* in wavelet transforms, deal with adapting the transform *on the fly* to the local properties of the signal, what implies an extra computational effort to detect its singularities. This classical meaning of adaptive lifting does not apply in this work. However, it refers to the adaptivity of the system as a whole. Therefore, adaptation becomes prior to system operation instead of real time adaptation to the signal at hand.

The structure of the paper is as follows. Sections II and III give an overview of both, DWT and ES. A brief analysis of previously reported works is given in Section IV. Afterwards, the proposed architecture is presented in Section VI, and a simplified ES directed towards an FPGA implementation in Section V. The experimental setup developed and a further insight into the details of the implementation, along with the results obtained can be found in Section VII. Paper is concluded in Section VIII.

II. Overview of the Discrete Wavelet Transform

The DWT is a multiresolution analysis (MRA) tool widely used in signal processing due to its joint time-frequency signal analysis characteristics, that concentrates the signal energy into fewer coefficients to increase the degree of compression when the data is encoded. For a general introduction to wavelet based MRA analysis see [3].

The *Lifting scheme* (LS), introduced by Sweldens [4], reduces the computational cost of the transform as required by the Fast Wavelet Transform (FWT) algorithm and facilitates the construction of custom wavelets for very specific and

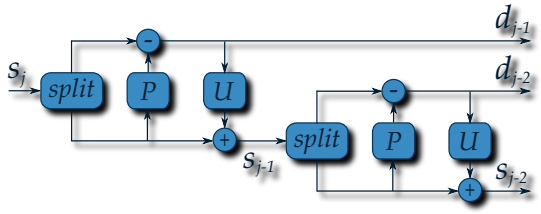

Fig. 1. Lifting scheme

different types of data. Besides, it is really well suited for the task of using an EA to encode wavelets, since any random combination of lifting steps will encode a valid wavelet, what guarantees perfect reconstruction [4], [5]. Fig. 1 shows the basic LS, which consists of three stages (also called lifting steps): ***Split*** (also called *Lazy Wavelet*, which simply divides the input data into even and odd samples), ***Predict*** and ***Update*** (convolution-like filters). The wavelet representation of s_j is given by the set of coefficients $\{s_{j-2}, d_{j-2}, d_{j-1}\}$. This scheme can be iterated up to n levels, so that an original input data set s_0 will have been replaced with the wavelet representation $\{s_{-n}, d_{-n}, \ldots, d_{-1}\}$. Therefore the algorithm for the LS implementation is:

for $j \leftarrow 1, n$ **do**
  $\{s_j, d_j\} \leftarrow Split(s_{j+1})$
  $d_j = d_j - P(s_j)$
  $s_j = s_j + U(d_j)$
end for

where j stands for the decomposition level. There exists a different notation for the transform coefficients $\{s_{j-i}, d_{j-i}\}$; for a 2 level image decomposition it is $\{LL, LH, HL, HH\}$ where L stands for low pass (data trend) and H for high pass (data details) coefficients respectively.

III. Bio-inspired optimization with Evolution Strategies

An Evolution Strategy (ES) is one of the fundamental algorithms among Evolutionary Algorithms (EA) that utilize a population of candidate solutions and bio-inspired operators to search for a target solution. ES is primarily used for optimization of real-valued vectors. The algorithm operators are iteratively applied within a loop, where each loop run is called a *generation* (g), until a termination criterion is met. Variation is accomplished by the so-called *mutation* operator. For real-valued search spaces, mutation is normally performed by adding a normally (Gaussian) distributed random value to each component under variation (i.e., to each parameter encoded in the individuals). Algorithm 1 shows a pseudo-code description of a typical ES.

The canonical versions of the ES are denoted by $(\mu/\rho, \lambda)$-ES and $(\mu/\rho + \lambda)$-ES, where μ denotes the number of parents (parent population, P_μ), $\rho \leq \mu$ the mixing number (i.e., the number of parents involved in the procreation of an offspring), and λ the number of offspring (offspring population, P_λ). The parents are *deterministically selected* from the set of either the offspring, referred to as *comma-selection* $(\mu < \lambda)$, or both the parents and offspring, referred to as *plus-selection*. Selection is based on the ranking of the individuals' fitness $(\mathcal{F})$ taking the μ best individuals. Once selected, ρ out of the μ parents $(\mathcal{R})$ are *recombined* to produce an offspring individual $(\mathbf{r_1})$ using *intermediate recombination*, where the parameters of the selected parents are averaged, or randomly chosen if *discrete recombination* is used. Each ES individual $\mathbf{a} := (\mathbf{y}, \mathbf{s})$ comprises the *object parameter vector* $\mathbf{y}$ to be optimized and a set of strategy parameters $\mathbf{s}$, which coevolve (and are therefore being adapted themselves) with the solution. This is a particular feature of ES called self-adaptation. For a general description of the $(\mu/\rho \overset{+}{,} \lambda)$-ES see [2].

Algorithm 1 $(\mu/\rho \overset{+}{,} \lambda)$-ES

1: $g \leftarrow 0$
2: Initialize $P_\mu^{(g)} \leftarrow \{(\mathbf{y_m}, \mathbf{s_m}), m = 1, \ldots, \mu\}$
3: Evaluate $P_\mu^{(g)}$
4: **while** not_termination_condition **do**
5: **for all** $l \in \lambda$ **do**
6: $\mathcal{R} \leftarrow$ Draw ρ parents from $P_\mu^{(g)}$
7: $\mathbf{r_1} \leftarrow recombine\,(\mathcal{R})$
8: $(\tilde{\mathbf{y}}_1, \tilde{\mathbf{s}}_1) \leftarrow mutate\,(\mathbf{r_1})$
9: $\mathcal{F}_l \leftarrow evaluate\,(\tilde{\mathbf{y}}_1)$
10: **end for**
11: $P_\lambda^{(g)} \leftarrow \{(\mathbf{y_1}, \mathbf{s_1}), l = 1, \ldots, \lambda\}$
12: $P_\mu^{(g+1)} \leftarrow selection\left(P_\lambda^{(g)}, P_\mu^{(g)}, \mu, \overset{+}{,}\right)$
13: $g \leftarrow g + 1$
14: **end while**

IV. Previous work on evolutionary wavelets design

This work is a continuation of [6], which uses the original idea of combining the lifting technique with EA for designing wavelets proposed by Grasemann and Miikkulainen [7]. Their original contributions are two: the use of a Genetic Algorithm (GA) to encode wavelets as a sequence of lifting steps and the proposal of an idealized version of a transform coder to save time in the complex evaluation method used (which involved computing a number of times the Peak Signal to Noise Ratio (PSNR) for one individual combined with other individuals from each of one of the parallel subpopulations): they propose using only a certain percentage of the largest coefficients (which involves a previous ordering stage) for reconstruction.

The GA used had parallel evolving populations (coevolutionary GA). The evaluation consisted of 80 runs, each of which took approximately 45 minutes on a 3 GHz Xeon processor. The results obtained in this work outperformed the considered state-of-the-art wavelet for fingerprint image compression, the D9/7 wavelet used by the FBI, in 0.75 dB. The type of images used to adapt the wavelet to is the set of 80 images from the FVC2000 fingerprint verification competition [8].

Works reported by Babb, Moore et. al. can be considered the current state of the art in the use of EC for image transform design [9], [10]. After using a GA algorithm in their previous works, the authors finally propose the use

of a ES, outperforming their previous results, but keep on encoding wavelets as filters for the FWT, instead of the LS. The milestones followed in their research are summarized on the next list.

1) Evolve the inverse transform for digital photographs under conditions subject to quantization.
2) Evolve *matched* forward and inverse transform pairs.
3) Evolve coefficients for three and four level MRA transforms.
4) Evolve a different set of coefficients for each level of MRA transforms.

The algorithms reported are highly computationally intensive (the training runs were done using the Arctic Region Supercomputer Center (ARSC), Fairbanks, Alaska). Although the work by Grasemann and Miikkulainen was done on an *accessible* computer, both training times and computing resources needed in both cases, show the complexity of the algorithms developed. These approaches are highly unfeasible for an implementation in an FPGA.

V. Proposed simplified ES for FPGA implementation

This work proposes using an ES as the search algorithm encoding the individuals (wavelets) using the LS. This is a combination of the original proposals analysed in Section IV. However, a standard ES is highly computationally intensive for an FPGA implementation, as these analysed works show. Therefore, the whole evolutionary process has to be downscaled in complexity. In the decisions made to simplify the algorithm are presented. They are summarized here:

1) *Single evolving population* opposed to the parallel populations of the coevolutionary genetic algorithm reported in [7] .
2) Use of *uncorrelated mutations with one step size* [2] instead of the overcomplex method reported in [9], [10].
3) Evolution of *one single set of coefficients for all MRA levels*.
4) *Ideal evaluation of the transform*. Since doing a complete compression would turn to be an unsustainable amount of computing time, the simplified evaluation method reported in [7] was further improved. For this work, all wavelet coefficients d_j are zeroed, keeping only the trend level s_j of the transform from the last iteration of the algorithm. For 2 levels of decomposition, this is an idealized 16:1 compression ratio.

These simplifications yielded very positive results, but, since there were still some complex operations around in the algorithm, the complexity relaxation before doing the hardware implementation was taken even further, so that a trade-off between performance and size is observed. They are summarized below:

1) *Uniform random distribution*. Instead of using Gaussian distributions for the mutation of the object parameters, a Uniform distribution was tested, for being simpler to implement in HW.

TABLE I
HW/SW partitioning of the system

EA Operator	Further actions	HW	SW
recombination	—		✓
mutation	—		✓
evaluation	*wavelet transform*	✓	
	fitness computation	✓	
selection	*sorting population*	✓	
	create parent population		✓

2) *MAE as evaluation figure*. PSNR is the figure of merit more widely used for image processing tasks. But, as previous works show for image filter design problems [11], using MAE gives almost identical results, because the interest is in relative comparisons among population members.

VI. Architecture mapping. HW/SW partitioning

Typical implementations of evolutionary optimization engines in FPGAs place the EA in an embedded processor. With this approach, some degree of performance is sacrificed to gain flexibility in the system (needed to fine tuning the algorithm), so that modifications may be easily done to the (software) implementation of the EA (which is, of course, much easier than changing its hardware counterpart). Table I shows the partitioning resulting from applying this design philosophy. According to Algorithm 1, each of the EA operators are shown in the table together with further actions to be accomplished: *recombination* (of the selected parents), *mutation* (of the recombinant individuals to build up a new offspring population), *evaluation* (of each offspring individual) and *selection* (of the new parent population).

When a new offspring population is ready, each of its individuals is sequentially sent to the hardware module responsible of its evaluation. This comprises the computation of the fitness as the Mean Absolute Error (MAE) as shown in (1), where R, C are the rows and columns of the image and I, K the original and transformed images respectively. To tackle it, the following sequence of operations has to be done: Forward Wavelet Transform (fWT), Compression (C), Inverse Wavelet Transform (iWT) (*wavelet transform*) and MAE figure computation (*fitness computation*). Once each offspring individual has been evaluated, the population is sorted according to the result (*sorting population*). At this stage, the microprocessor may close the evolutionary loop creating the new parent population. Afterwards, recombination and mutation are applied and a new offspring population will be available to be evaluated.

$$MAE = \frac{1}{RC} \sum_{i=0}^{R-1} \sum_{j=0}^{C-1} |I(i,j) - K(i,j)| \qquad (1)$$

Fig. 2 shows the proposed conceptual architecture capable of hosting such a system. The functions to be implemented in hardware work as attached peripherals to the microprocessor embedded in the FPGA.

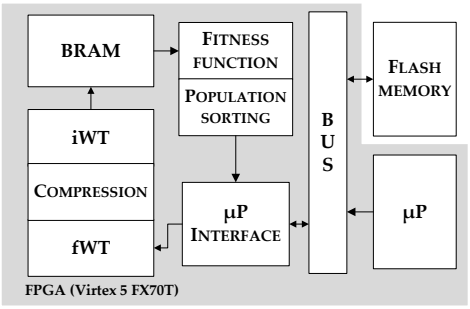

Fig. 2. System level architecture

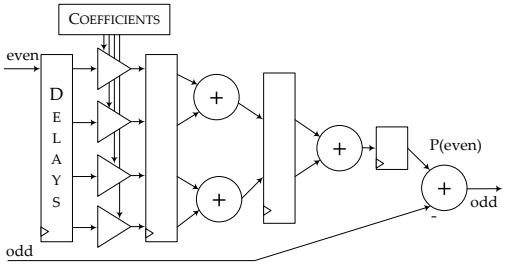

Fig. 3. Predict/Update stage implementation

TABLE II
PROPOSED EVOLUTION STRATEGY

Parameter / Operator	Value
Representation	$\langle x_1, \ldots, x_n, \sigma \rangle$ $n = 26$, fixed point coefficients
Wavelet Encoding	$\langle P_1, U_1, P_2, U_2, P_3, U_3, k_1, k_2 \rangle$
Mutation [ab]	$\sigma' = \sigma \cdot \exp^{\tau \cdot N(0,1)}$ $x'_i = x_i + \sigma' \cdot U_i(-\sigma', \sigma')$
Learning rate τ	$\tau \propto 1/\sqrt{\alpha n}, \alpha = \{1, 2\}$
Evaluation	MAE
Selection	Comma
Recombination	Intermediate, $\rho = 5$
Parent population size	$\mu = 10$
Offspring population size	$\lambda = 70$
Seed for initial population	Random and D9/7

[a] $N(0, 1)$: draw from the standard normal distribution
[b] $U_i(-\sigma', \sigma')$: separate draw from the discrete uniform distribution for each variable i

Since the LS was proposed, several hardware implementations have been reported both for ASICs and FPGAs (JPEG2000 adopted LS). This means that good results centred on exploiting LS features to obtain fast implementations have already been done. But the objectives at this stage of the work are just directed to prove and validate the concepts and the feasibility of the system as a whole. Therefore, the implementation of the Wavelet Transform is a direct, algorithmic mapping of the LS to its hardware equivalent VHDL description (i.e., no hardware optimizations at the level of data dependencies are accomplished).

Taking advantage of the LS features, the fWT and iWT can be computed by just doing a sign flip and a reversal in the order of operations (P and U stages), so both modules are sharing hardware resources in the FPGA. Compression block is simple, since it only needs to substitute the fWT result by zeros for each datum of the details bands. Therefore, it is working in parallel to the fWT. In a similar manner, the Fitness module computes the difference image as each pixel is produced by the iWT.

The fWT/iWT module is build up by applying the sequence of P, U stages dictated by the LS. To mimic the high level modelling of the algorithm (see Section VII), 6 stages have been implemented (3 P and 3 U), each one containing 4 filter coefficients, as seen in Fig. 3, which is enough to implement the most common wavelets utilized at present. Section VII shows the first preliminary results of the implementation.

The Block RAM modules (BRAM) embedded in the FPGA are used as data memory for the wavelet transform module. During evolution, it hosts the training image so that the highest memory bandwidth possible is achieved. It has been over-dimensioned to host up to 4 256x256 pixels (8 bpp) images to speed up the test phase. Therefore, when evolution has finished, this extra memory can be used to load the test images from system memory. In this phase, one of the four sub-banks is used for the actual image being tested, and the other three are loaded with the following test images in the meanwhile, acting as a multi ping-pong memory.

VII. EXPERIMENTAL SETUP AND RESULTS

Before accomplishing the hardware implementation, modelling and simulation of the algorithm was done with Python computing language [12], together with its numerical and scientific extensions, NumPy and Scipy [13], as well as the plotting library, MatPlotlib [14].

As shown on [15], [16], for 8 bits per pixel (bpp) integer inputs from an image, a fixed point fractional format of Q2.10 for the lifting coefficients and a bit length in between 10 and 13 bits for a 2 to 5 level MRA transform for the partial results is enough to keep a rate-distortion performance almost equal to what is achieved with floating point arithmetic. Therefore, fixed-point binary arithmetic (16 bits fractional part) was modelled with integer types, defining the required quantization/dequantization and bit-alignment routines to mimic hardware behaviour. Table II gathers all the information related to the proposed ES, where each P_i, U_i is made up of 4 coefficients and k_i are single scaling coefficients. D9/7 wavelet is defined by $\langle P_1, U_1, P_2, U_2, k_1, k_2 \rangle$.

To sum up, for the various tests run, the set of parameters used correspond to a $(10/5, 70)$-ES with a varying initial $\sigma = \{0.1, \ldots, 1.5\}$.

The experiments reported in this work have also used, as in [7], the first set of 80 images of the FVC2000 fingerprint verification competition. Images were black and white, sized 300x300 pixels at 500 dpi resolution. One random image was used for training and the other 79 for testing purposes.

A. Algorithm optimization results

All the results obtained are compared with the D9/7 transform implemented in fixed point arithmetic and evaluated with

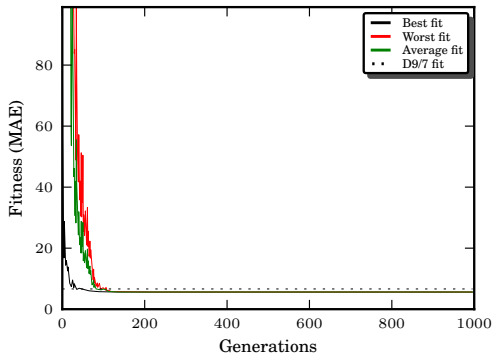

Fig. 4. Result of a typical evolution run

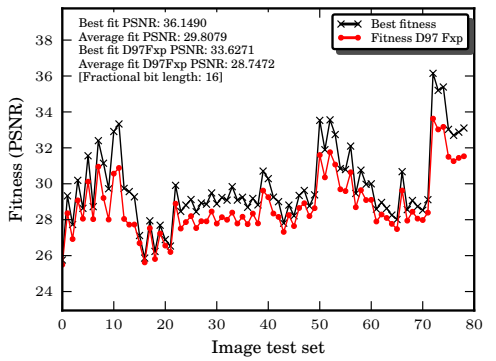

Fig. 5. Tests of the best evolved wavelet. The best individual (and D9/7 for comparison) is exercised for each of the 79 images of the test set

the proposed method. Although evolution used MAE as the fitness function, for comparison purposes, the results shown in this Section are given as the standard PSNR. As a reference, for the fixed point D9/7 wavelet the average PSNR measured (with the proposed evaluation method) for the 79 test images is 28.74 dB.

Fig. 4 shows how the algorithm behaves during a typical run. As it can be seen, around generation 100 the performance of the evolved wavelet is already similar to the D9/7. In Fig. 5 the best evolved wavelet is compared with the D9/7 for the whole image test set, showing how the algorithm was able to evolve a solution yielding 29.80 dB, that outperforms the standard D9/7 by an average of 1,06 dB. Results correspond to $\sigma = 1.0$ and a randomly seeded population. Fig. 6 shows a comparison of the performance of the best evolved wavelet against a fixed point implementation of D9/7 for a test image, where some artefacts can be noticed in the D9/7 result. Besides the error images show how the error introduced by the best evolved wavelet is lower, indicating a more accurate reconstruction of the original image.

B. HW/SW partitioning validation

The model developed to validate the algorithm has been profiled. Table III shows profiling results for 500 generations for each EA operator. Table I is repeated (for clarity) adding extra columns with the result values. Absolute values are not of

TABLE III
ALGORITHM CODE PROFILING

EA Operator	Further actions	HW	SW	Time[a]	%
recombination	—		✓	0.14	0.009
mutation	—		✓	0.43	0.029
evaluation	*wavelet transform*[b]	✓		1433.56	97.470
	compression	✓		4.96	0.337
	fitness computation	✓		31.62	2.150
selection	*sorting population*	✓		0.040	0.003
	parent population		✓		

[a] All results in seconds
[b] Results show computation time for both, forward and inverse wavelet transform

real interest (although NumPy routines are highly optimized, a C implementation would be faster), since what is being checked is the relative amount of time spent in each phase so that design partitioning is validated as a whole. As expected, most of the time is consumed evaluating the individuals. In each generation, 20.479 ms $(= 1433.56/(500 \text{ generations} * 70 \text{ individuals} * 2 \text{ transforms}))$ are needed to compute a single wavelet transform (forward or inverse). Obtained results validate the design partitioning proposed except for the *selection* operator, which is low enough to be implemented in SW. The reason to choose a HW implementation for it, is that it can be applied as results are produced by the fitness computation module, saving extra time. In contrast, the simulation of the Python model runs on a single processor thread. Therefore, all operators are applied sequentially. But in the hardware implementation, some operators can be easily applied in parallel. For this reason, and depending on the scope of the system (see Section VIII), some other operator will probably benefit from being implemented in hardware, as, for example, the mutation. Besides, the subset of the C-language used to program the PowerPC processor in the FPGA, impose restrictions that will probably make that the percentage of the time each operator takes to compute increases.

C. Preliminary Hardware results

The prototype platform selected is an ML507 development board, which contains a Xilinx Virtex 5 XC5VFX70T FPGA device with an embedded PowerPC processor, responsible of running the ES. Table IV shows the preliminary implementation results for an over-dimensioned datapath of 32 bits, using 16 bits for the fractional part representation. This implementation is directed towards a system level functional validation in the FPGA, yielding higher area results than expected for the final system.

The current hardware, non-optimized, implementation yields one result each clock cycle. For a 256x256 pixels image, with a clock frequency of 100 MHz, the computation time of a wavelet transform is approximately 0.65 ms. This is a speed-up factor of around $(20.5/0.65)$ 31 times, what would turn into 45 seconds to do all the transforms required by a population of 70 individuals during 500 generations.

(a) (b) (c) (d) (e)

Fig. 6. Transform performance. (a) is the original fingerprint image; (b) shows the result of applying the D9/7 Fixed Point implementation wavelet and (c) the equivalent for the best evolved individual; (d) shows the difference image for the D9/7 result and (e) for the best evolved wavelet.

TABLE IV
PRELIMINARY IMPLEMENTATION RESULTS FOR THE MAIN MODULES IN THE SYSTEM.

Module	Resources	Frequency
fWT/iWT	3651 / 11200 (12%)	181 MHz
Fitness function	25 / 11200 (<1%)	330 MHz
Population sorting	984 / 11,200 (8%)	265 MHz

VIII. DISCUSSION AND FUTURE WORK

A simplified ES to evolve wavelets has been proposed and validated using fixed-point arithmetic, outperforming the current state of the art D9/7 transform (used by FBI in its compression standard). The profiling results of the algorithm modelling validates the proposed HW/SW partitioning. The resulting hardware architecture is being implemented on an FPGA device. A preliminary test implementation has been accomplished to perform a system level functional validation. As preliminary synthesis results show, the FPGA will be able to host the complete system.

Currently, the rest of the system is being implemented before functional simulations are done and an optimized version is implemented if needed. In parallel, further simplifications to the ES (specifically to the mutation operator) are being tested, so that a simpler implementation is possible.

The current status of this work shows how adaptive compression for embedded systems based on bio-inspired algorithms can be faced. Besides, since the process is sped-up by a large factor in the hardware implementation as compared to the software, PC-based model, the system can also be conceived as an accelerator for the optimization process of wavelet transforms (for the construction of custom wavelets).

ACKNOWLEDGEMENT

This work was supported by the Spanish Ministry of Science and Research under the project DR.SIMON (Dynamic Reconfigurability for Scalability in Multimedia Oriented Networks) with number TEC2008-06486-C02-01.

Lukas Sekanina has been supported by MSMT under research program MSM0021630528 and by the grant of the Czech Science Foundation GP103/10/1517.

REFERENCES

[1] S. Mallat, *A Wavelet Tour of Signal Processing, Second Edition*, 2nd ed. Academic Press, Sep. 1999.

[2] H. Beyer and H. Schwefel, "Evolution Strategies. A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, Mar. 2002.

[3] B. Jawerth and W. Sweldens, "An overview of wavelet based multiresolution analyses," *SIAM Review*, vol. 36, no. 3, pp. 377–412, 1994.

[4] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186 – 200, 1996.

[5] ——, "The lifting scheme: a construction of second generation wavelets," *SIAM J. Math. Anal.*, vol. 29, no. 2, pp. 511–546, 1998.

[6] R. Salvador, F. Moreno, T. Riesgo, and L. Sekanina, "Evolutionary design and optimization of wavelet transforms for image compression in embedded systems," in *Proc. of the 2010 NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE Computer Society, 2010, pp. 177–184.

[7] U. Grasemann and R. Miikkulainen, "Effective image compression using evolved wavelets," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*. Washington DC, USA: ACM, 2005, pp. 1961–1968.

[8] D. Maio, D. Maltoni, R. Cappelli, J. Wayman, and A. Jain, "FVC2000: fingerprint verification competition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 3, pp. 402–412, Mar 2002.

[9] B. J. Babb, F. W. Moore, and M. R. Peterson, "Improved multiresolution analysis transforms for satellite image compression and reconstruction using evolution strategies," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. Montreal, Qubec, Canada: ACM, 2009, pp. 2547–2552.

[10] B. Babb, F. Moore, M. Peterson, T. H. O'Donnell, M. Blowers, and K. L. Priddy, "Optimized satellite image compression and reconstruction via evolution strategies," in *Evolutionary and Bio-Inspired Computation: Theory and Applications III*, vol. 7347. Orlando, FL, USA: SPIE, May 2009, pp. 73 470O–10.

[11] Z. Vasicek and L. Sekanina, "An evolvable hardware system in Xilinx Virtex II Pro FPGA," *Int. J. Innov. Comput. Appl.*, vol. 1, no. 1, pp. 63–73, 2007.

[12] G. V. Rossum, *The Python Language Reference Manual*. Network Theory Ltd., Sep. 2003.

[13] T. E. Oliphant, "Python for scientific computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 10–20, 2007. [Online]. Available: http://link.aip.org/link/?CSX/9/10/1

[14] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[15] M. Martina, G. Masera, G. Piccinini, and M. Zamboni, "A VLSI architecture for IWT (integer wavelet transform)," in *Circuits and Systems, 2000. Proceedings of the 43rd IEEE Midwest Symposium on*, vol. 3, 2000, pp. 1174–1177 vol.3.

[16] M. Grangetto, E. Magli, M. Martina, and G. Olmo, "Optimization and implementation of the integer wavelet transform for image coding," *Image Processing, IEEE Transactions on*, vol. 11, no. 6, pp. 596–604, 2002.